

This document was prepared in conjunction with work accomplished under Contract No. DE-AC09-96SR18500 with the U. S. Department of Energy.

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

This report has been reproduced directly from the best available copy.

**Available for sale to the public, in paper, from: U.S. Department of Commerce, National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161,
phone: (800) 553-6847,
fax: (703) 605-6900
email: orders@ntis.fedworld.gov
online ordering: <http://www.ntis.gov/help/index.asp>**

**Available electronically at <http://www.osti.gov/bridge>
Available for a processing fee to U.S. Department of Energy and its contractors, in paper, from: U.S. Department of Energy, Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831-0062,
phone: (865)576-8401,
fax: (865)576-5728
email: reports@adonis.osti.gov**

Integrating New Technology Solutions to Improve Plant Operations

Eric J. Heavin
Engineer
Bechtel Savannah River Inc.
Savannah River Site
Aiken, SC 29808

Lance Abbott
Principal Engineer
Westinghouse Savannah River Co.
Savannah River Site
Aiken, SC 29808

Daniel Shanyfelt
Engineer
Bechtel Savannah River Inc.
Savannah River Site
Aiken, SC 29808

KEYWORDS

Distributed Control System (DCS), Software, Application Programming Interface (API), Rapid Application Development (RAD), Visual Studio .NET, Data Modeling

ABSTRACT

Continuing advancements in software and hardware technology are providing facilities the opportunity for improvements in the areas of safety, regulatory compliance, administrative control, data collection, and reporting. Implementing these changes to improve plant operating efficiency can also create many challenges which include but are not limited to: justifying cost, planning for scalability, implementing applications across varied platforms, integrating multitudes of proprietary vendor applications, and creating a common vision for diverse process improvement projects.

The Defense Programs (DP) facility at the Savannah River Site meets these challenges on a daily basis. Like many other plants, DP, has room for improvement when it comes to effective and clear communication, data entry, data storage, and system integration. Specific examples of areas targeted for improvement include: shift turnover meetings using system status data one to two hours old, lockouts and alarm inhibits performed on points on the Distributed Control System (DCS) and tracked in a paper logbook, disconnected systems preventing preemptive correction of regulatory compliance issues, and countless examples of additional task and data duplication on independent systems.

Investment of time, money, and careful planning addressing these issues are already providing returns in the form of increased efficiency, improved plant tracking and reduced cost of implementing the next process improvement.

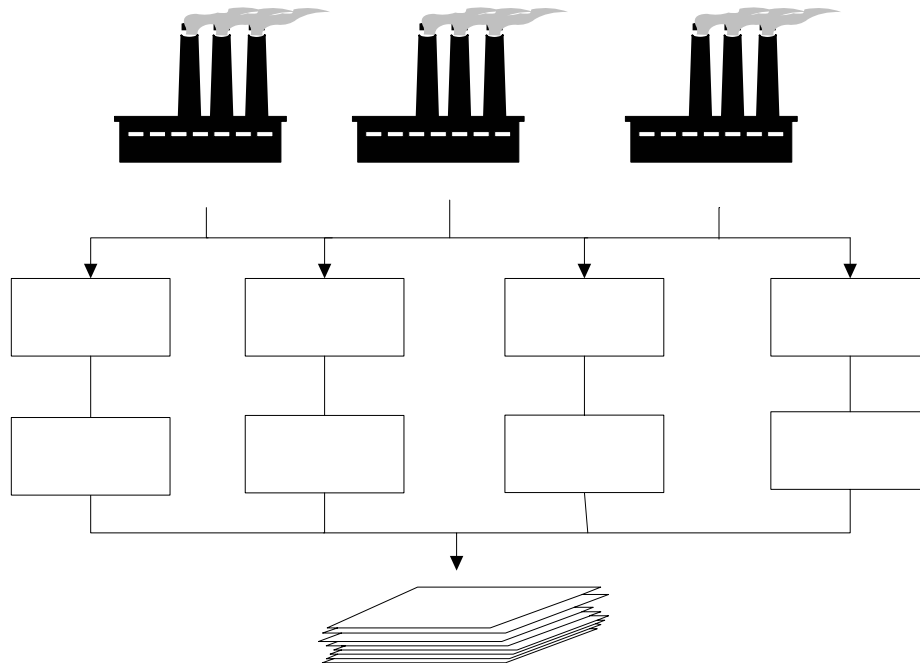
Specific examples of improving plant operations through thoroughly planned Rapid Application Development of new applications are discussed. Integration of dissimilar and independent data sources (NovaTech D/3 DCS, SQL Server, Access, Filemaker Pro, etc.) is also explored. The tangible benefits of the implementation of the different programs to solve the operational problems previously described are analyzed in an in-depth and comparative manner.

INTRODUCTION

Technology offers industrial applications near-limitless opportunities to increase efficiency, improve regulatory compliance, expand product lines, and increase quality control. Every process from the production line, to human resources, to the most mundane administrative task, is a candidate for improvement through the utilization of the latest technological solutions. While the possibilities are endless, implementing these process improvements can be a daunting and costly task. Many facilities have become risk averse after a history of poorly planned projects promising amazing results, but often delivering products that are expensive, difficult to maintain, impossible to upgrade and fall far short of their promised value. Careful planning, however, can lay a foundation upon which “process engineering” groups can implement solutions in a cost-effective way without having to become an information technology (IT) group. This paper provides an overview of how to create this broad technology plan along with examples of where this plan benefited development at the Savannah River Site DP facility, and where holes in this plan led to extra work and cost.

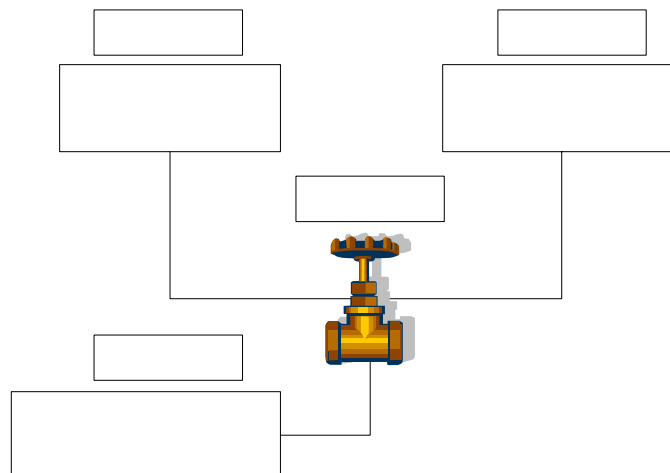
SYSTEM ANALYSIS AND ARCHITECTURE

To create an effective technology plan, engineers must first analyze their system as a whole and learn how to model that system using some of the fundamentals of modern software design. The initial analysis must be comprehensive. That means looking outside of the areas that most systems engineering groups are familiar with and determining how the process system relates to all of the support and administrative systems used to operate the facility. Starting from a major key component in one’s system and looking at all of its supporting and relational entities is crucial to determining data flow for that system and outer system involvement. Figure 1 is an example of an overall analysis of the DP operations facility displaying key components that contribute in generating a daily facility operations report.



In Figure 1, all of the different components supply data to the “facility report” by means of facility work performed, and the direct output from the process. This first big-picture look at all of the related components of the system allows the model to be broken down by system for more detailed analysis by those groups or individuals most intimately familiar with their operation. Breaking down individual model components allows one to determine items such as: the source of the data, programs and software used, and actions or work performed (see Figure 2).

Proc



Operations

Product

Performing a thorough system analysis is not only an essential step in software modeling, but also helps identify targets for improvement by providing a baseline against which improvements can be judged for cost and potential return..

Once the system components have been analyzed, software engineering solutions can be approached. For the software engineering portion, programming and software design expertise is not required. However, a solid comprehension of Object Oriented Programming (OOP), Data Modeling and n-tier solution design are essential to the success of even the smallest hardware or software project. According to Evangelos Petoutsos and Asli Bilgin, “Any well-designed distributed system should acknowledge four important points, which we refer to as the “abilities” of an application: Interoperability, Scalability, Reusability, and Extensibility”¹. These fundamentals are the tools needed to the turn system analysis into the “technology plan” where individual projects of any scope can be developed with the confidence that they will integrate with and support the next project in the pipe as well as future projects. This kind of interoperability will drastically improve return on investment as the tools lifespan is increased and maintenance costs reduced.

DESIRED MODIFICATIONS

Even after a thorough system analysis it still may not be obvious which areas to target for improvement first. Obvious first choices include the following:

- Related functionality spread across multiple systems
- Important data not collected or difficult to retrieve
- Confusing user interfaces

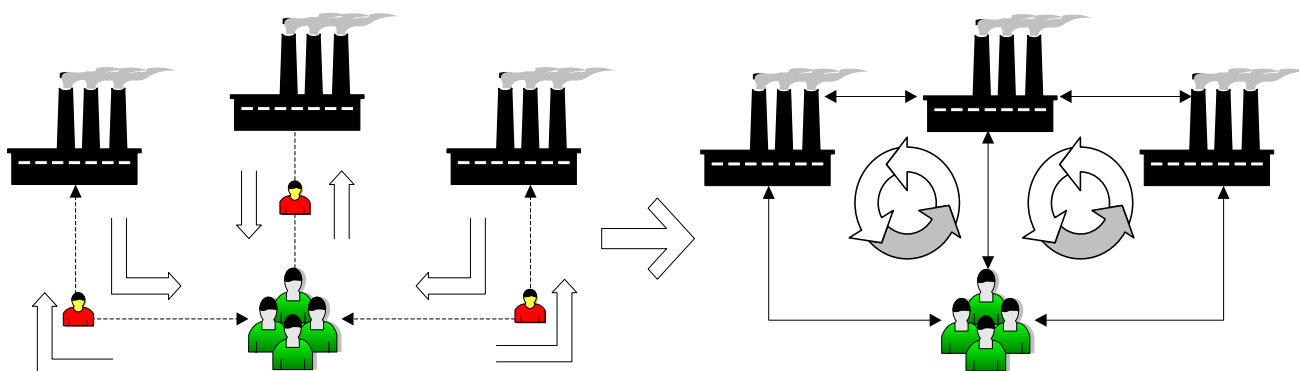
Most of these were apparent before the system analysis, so what has this work really done to clarify the situation? First, a better picture of approaching system modification tasks and how much effort is involved in solving these problems is starting to form. Second, a number of more subtle, yet valuable opportunities are now available for consideration. Some targets that may be less obvious, but often present great opportunities for cost-effective improvement include the following:

- Data duplication
- Broad-scope reporting
- Functionality Duplication

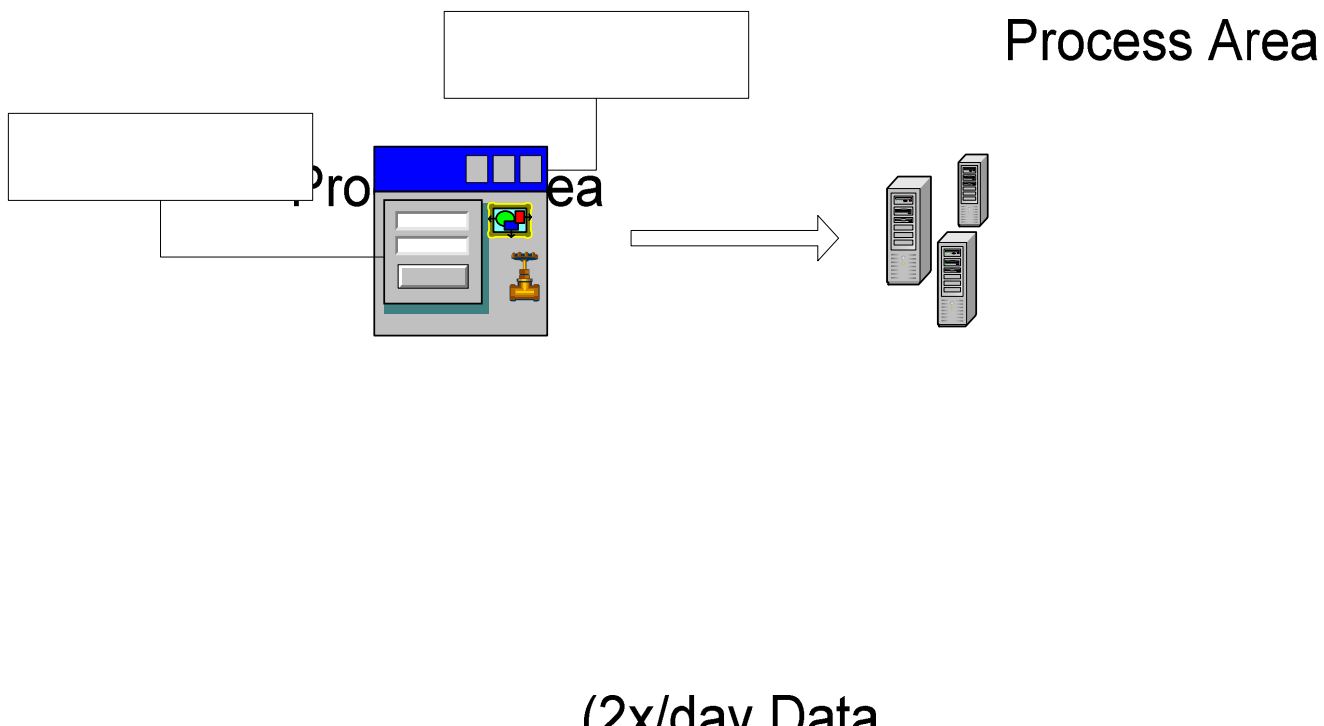
There were two main projects that drove the initial DP system analysis and architecture configuration. These projects will be referred to as the Facility Status Board (FSB), and the System Status Log (SSL). Both were inspired by the need to enhance facility communication, eliminate data duplication, and reduce “operator error” by being able to implement better administrative controls. These projects will be chronicled throughout the remainder of this paper to help share some of the “lessons learned” in a facility with the desire to implement the newest technology to solve every problem, but without the resources to do so without careful planning and expanding the skills of process engineers into the realm of IT.

The **FSB** project was driven by the inefficiency of communication between on-coming and off-going shifts. There are several different buildings and systems that contribute to the shift turnover meetings. Each area was required to bring data from the separate control rooms, offices, and work areas on

printouts to one central meeting room. The system analysis revealed (in Figure 1) several different input streams with duplicate data, isolated networks, and data validity issues. The solution for this project (see Figure 3) was to design or obtain a program that could be run and updated from multiple locations throughout the facility on a single accessible network for determining current facility status information. Each section of the facility was then divided up into different electronic “boards” of data where their equipment and area information could be updated for display to the rest of the plant.



The SSL project was created to meet two critical facility needs. One of these needs was to provide operations a tool to make large system status changes (lockouts, alarm inhibits, caution tags) of groups of points instead of a single point at a time. The second facility need was to provide an electronic log to improve accountability and provide easily accessible information of system configuration. The system analysis (see previous example in Figure 2) revealed the inconsistency of operators having to log system status changes performed on the DCS in separate log sheets or databases isolated from the DCS, and operators losing valuable time performing mass system status changes. The answer to this dilemma was to design or obtain a program that could add all the administrative controls and tracking information of performing a system status change to a tool that the operator could use on the DCS.



RESOURCE ALLOCATION

After laying the groundwork for the software solution and creating a proposal for development, much thought should be given to appropriately allocating resources to different project tasks. The task of data modeling should be performed by a knowledgeable modeler who is in contact with the different area “experts” for the proposed model. The initial design should be as all-encompassing as possible. A good interviewing practice to follow for the data modeler would be to ask each process area expert about their system requirements, but also ask questions concerning expansion, frequency, and any special cases to allow for more design room and growth. Open-ended questions like “Do you think you would utilize this addition” or “Would it make sense to if this was performed this way” invokes thought into what is trying to be done, and involves and challenges the area expert to take a different look at his/her current setup.

Interface design of an application should at least take into consideration comments of as many developers, designers, and users as possible. Interfaces should be as standardized as possible to minimize user and operator retraining during revisions or modifications, and to assist in the goal of developing applications in modular form (see User Interface section for continued discussion).

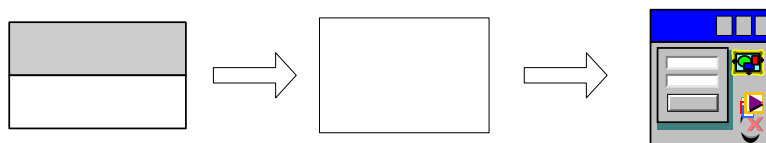
DEVELOPING A DATA MODEL

The most critical part of creating architecture, the data model, is often completely ignored or paid very little attention to. A well designed data model will outlast every software package in a plant, developed or acquired. A poorly designed data model, on the other hand, will increase installation cost, complicate maintenance and shorten the lifespan of those same packages. Changes to a data model currently in use can and will require rework of every referencing program. Investing time and effort to understand the all of the systems in the plant and how to represent them in a way that is complete, useful and flexible will give greater returns than any other phase of development.

How does a system engineering group go about creating a data model? A discussion of the details of data modeling and how to create and manage relational databases is far beyond the scope of this paper. Many resources, however, make the field accessible to the amateur IT engineer. *Data Modeling Essentials*² is highly recommended. What is within the scope of this paper however, is to further the previously mentioned case studies of the FSB and SSL projects as they pertain to data modeling. An important point to mention here is the fact that the data model for these two projects are distinct only because of a particular data isolation (isolated network due to classification) issue encountered as a result of the type of data that is handled in the DP Facility. Unless there is a compelling reason not to (security, data integrity, performance, etc.), all facility data should be represented using a single model. Even if this model must be replicated on different isolated networks.

For the FSB program, the initial data model consisted of having status boards created by the program directly bound to individual tables within the database. Each table was designed based what was desired to be displayed on its own particular status board (see Figure 5). One of the first problems

encountered with this model was how to display data on multiple boards without duplicating the data in the DB.



Basing the FSB application design so directly on the structure of the database made changes to either entity very difficult and time consuming, with extensive testing required.

The newer FSB database model allowed for as much expansion or reduction as envisioned possible. This was accomplished by making a completely relational database with the ability to have multiple areas with dependent systems, and pieces of equipment (see Figure 6).

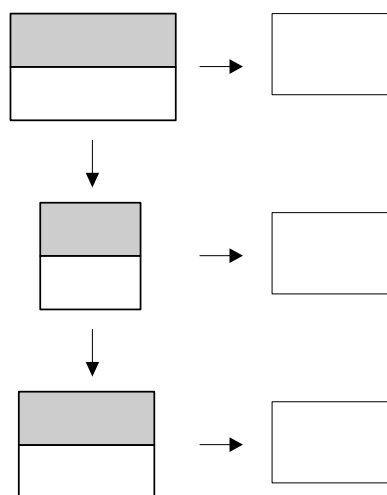
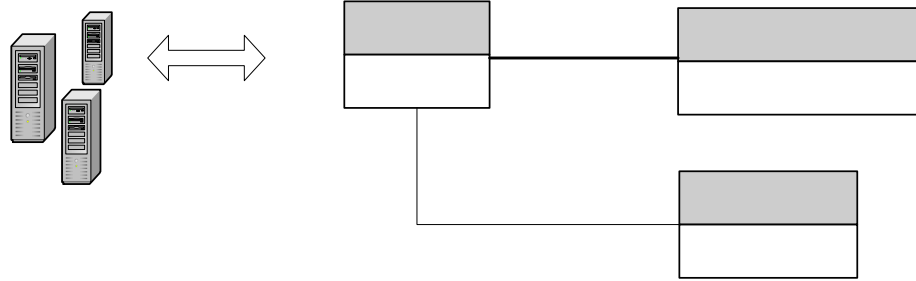


Table
System Stat

The newer model design was based on the actual facility data architecture, rather than what was desired to be displayed on an individual status board. Using this structure made this data readily available for access by any type of program. This design also allowed for a much more versatile and distributable database, which could not only be applied to the DP facility, but any other process facility.

The SSL data model was designed to replicate a relational version of the “flat file” DCS Continuous Database (CDB) in order to track, log, and perform the required system status changes. This database was designed to seamlessly integrate with the DCS (see Figure 7).



The initial data model for the SSL program was very good at having the most current system information from the DCS, but had several issues with keeping historical data. This mainly dealt with the ability to update DCS information (point names, type, etc.) that were tracked within the different status logs. Since the database design called for a force-update table relationship with the DCS, all historically tracked points would be updated with the most current DCS information even if that wasn't the value at the time it was entered. To be able to keep true historical data, the data model needed to be modified to allow for DCS point revisions.

As one can gather from reading the previous examples, many different designs of different data models were analyzed during the development process. After multiple revisions it became apparent that while there were no entirely "perfect" data models, there were clearly models that better met the requirements than others. All models are compromises between the principles of data integrity, normalization, performance and functionality. The key to building a solid data model is understanding these compromises, and making the choice that best suits the needs of the facility.

USER INTERFACES

A good user interface can make, or break a potentially brilliant application. Creating applications with a standardized and intuitive design increases the odds of user acceptance of the software. Adhering to a set of Human Factors standards³ helps ensure a consistent "look and feel" among existing and new utilities. Experience in the DP Facility has demonstrated that end users identify more with how they relate with a product than what functionality the product supplies. Therefore, the consistent look and feel of new or modified applications greatly increases user comfort. Time and effort spent standardizing interface logic is easily recaptured with savings in training and support. Creating standard objects and controls to use for the same functions within separate applications enhances the ability to re-use controls, forms, and program layouts.

No one knows more about a particular end user's job than the end user them self. Each new product initially decreases the user's overall familiarity with how to perform that task. For successful application acceptance and implementation, the user must recognize how the product benefits them

when it is introduced. As previously mentioned, using RAD techniques to familiarize the end user with the product and collect continuous feedback during the early stages of development are both critical to integrating new applications in a continuously operating process environment. Taking the time to give users more input into the application contributes to a feeling of ownership on the part of the end user. When the end user feels invested in the product, they are far more likely to view the product as a way to make their job easier than as another ill-conceived imposition by a process support group.

While important, user input must still be balanced with good development practices. Development of the interface must be a team effort between both developers and users. The user's initial ideas on how an interface should operate may prove to be inefficient or unusable in practice. Working with the user to guide them to a final product that both satisfies their needs and is based on solid design principle and HFE standards is a great example of a balanced development process.

CONCLUSION

Thorough planning and detailed design is essential in the success of any technology plan. Performing a detailed system analysis can not only better paint a picture of the effort of implementing new technological solutions into ones process, but also make known some of the less obvious system weaknesses.

For the DP Facility, thousands of man-hours were saved after the successful implementation of the FSB program. The facility also went from having two major process building's information readily available at shift turnover, to five. The SSL program opened the door for developing custom applications for better plant operation such as tracking and logging operating caution tags on devices, performing system maintenance, and obtaining detailed DCS shift reports.

REFERENCES

1. Petroutsos, Evangelos & Bilgin, Asli, Visual Basic .NET Database Programming, Sybex, © 2002
2. Simison, Graeme C., Data Modeling Essentials – Analysis Design, and Innovation – 2nd Edition, Coriolis © 2001
3. Office of Nuclear Regulatory Research, NUREG-0700: Human System Interface Design Review Guideline, ©1996
4. Charles Williams, Professional Visual Basic 6 Databases, Wrox Press Ltd. © 1999